Performance Analysis of the Double-Iterated Kalman Filter for Molecular Structure Estimation

D. DELFINI and C. NICOLINI*
Istituto di Biofisica, Università di Genova, Via Giotto 2, 16153 Genova, Italy

E. A. CARRARA
Cibernia s.r.l., Via B. D'Este 44, 20141 Milano, Italy

Received 14 February 1995; accepted 29 June 1995

ABSTRACT

A possible application of a novel double-iterated Kalman filter (DIKF) as an algorithm for molecular structure determination is investigated in this work. Unlike traditional optimization algorithms, the DIKF does not exploit experimental nuclear magnetic resonance (NMR) constraints in a penalty function to be minimized but used them to filter the atomic coordinates. Furthermore, it is a nonlinear Bayesian estimator able to handle the uncertainty in the experimental data and in the computed structures, represented as covariance matrices. The algorithm presented applies all constraints simultaneously, in contrast with DIKF algorithms for structure determination found in literature, which apply the constraints one at a time. The performances of both paradigms are tested and compared with those obtained by a commonly used optimization algorithm (based on the conjugate gradient method). Besides providing estimates of the conformational uncertainty directly in the final covariance matrix, DIKF algorithms appear to generate structures with a better stereochemistry and be able to work with realistically imprecise constraints, while time performances are strongly affected by the heavy matricial calculations they require.

Introduction

To address the problem of the three-dimensional structure determination of polypeptides from NMR experimental distance constraints, various algorithms have been developed, all involving the minimization of a penalty function: energy minimization in simulated annealing[1] and restrained molecular dynamics[2]; distance minimization in distance geometry algorithms[3]; and dihedral angle minimization (with

* Author to whom all correspondence should be addressed.

CCC 0192-8651 / 96 / 010074-13

a variable penalty function).[4] Other approaches exploit systematic conformation searches[5] and Monte Carlo simulations.[6] Nevertheless, none of these methods represents the ideal solution to the problem. Distance geometry is biased toward extended structures,[7] simulated annealing and minimization methods in general generate one structure at a time, and the goodness of the structure estimation depends on the homogeneity of the sampling of the conformational space produced by generating one structure at a time, with different starting structures. Since the relationship between the starting structure and the minimized one is rather loose and depends on factors that are not completely under the user's control (e.g., the precision of floating point operations in the computer), criteria for choosing starting structures to grant homogeneous sampling are not obvious. Furthermore, such methods may be trapped in local minima of the penalty function.

The Kalman filter[8-14] (in particular in its double-iterated form, double-iterated Kalman filter, DIKF) appears to be well suited to handle problems of this kind. In fact, it does not exploit any penalty function and thus avoids the related problems of choosing the suitable function and minimizing it without getting trapped in local minima. Furthermore, since it explicitly generates an estimation of atomic position uncertainties along with the structure, DIKF reduces the need for multiple runs to sample the conformational space and the impact of starting structure choice on the quality of sampling.

DIKF algorithms presented so far in this contest operate in a sequential mode—that is, with a one-by-one application of the constraints.[9-14] In this sequential mode, the matricial operations of the Kalman filter degenerate to a sequence of vector operations, making the result dependent on the order in which the constraints are applied. Our algorithm (described in the first paragraph) actually applies the constraints simultaneously, with fully matricial operations. Despite the arguments of some authors[13] that applying all the constraints simultaneously would be equivalent to applying them one at a time, the two algorithms are not mathematically equivalent. It is well known that the results produced by the sequential DIKF depend on the application order of the constraints, to such an extent that a constraint sorting step is part of sequential DIKF-based algorithms for structure determination.[9-14] The results produced by the parallel DIKF are order independent. In Appendix A we show a simple example (with three points and two distance constraints) of the nonequivalence of the two algorithms.

The performances of a sequential[12] and of the parallel DIKF have been compared with those obtained with the conjugate gradient method as implemented in the program Dspace 4.0, by Hare Research, Inc., under the same conditions (starting structures and constraints). The experimental data set is described in the section titled "The Test Set of Amino Acids" and commented on in the section titled "Results." Although most articles regarding algorithms for protein structure determination report tests on proteins, in which pseudoatoms or backbone representations are often used,[1-3] a rigorous characterization of such algorithms needs tests on simpler structures to gain better control of the computation parameters and to make possible a rigorous interpretation of the results. We thus chose to exploit tests on single amino acids, which appear to be well suited for this purpose: Besides being the basic units of proteins, they have also a limited number of atoms (no more than 27) and yet show some moieties (such as the aromatic rings or the branched sidechains) that are sufficiently complex to provide a first bench test for the program.

The Algorithm

The flow chart in Figure 1 refers to the parallel DIKF, referred to as PDIKF from now on to distinguish it from the sequential DIKF (SDIKF); summarizing lists of the symbols used are reported in Tables I and II. The PDIKF exploits a parametric representation of the solution space, called *map*; a map is organized in a $3N$-dimensional state vector $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$, containing the mean coordinates of the N points and in a covariance matrix $\mathbf{C}(3N \times 3N)$, containing the variances and the covariances of the coordinates. The M constraints are represented by the M-dimensional vector $\mathbf{z} = (z_1, z_2, \ldots, z_M)$ and by the covariance matrix $\boldsymbol{\nu}(M \times M)$, which is usually assumed to be diagonal (i.e., the constraints are assumed to be independent). Two indexes are necessary to identify the map at each step of the double-iterated computation: An index k as outer loop index (we call it cycle) and an index i as inner loop index (we call it iteration). At the ith iteration of the kth cycle the state vector is updated with all the constraints,

Double-Iterated Kalman Filter

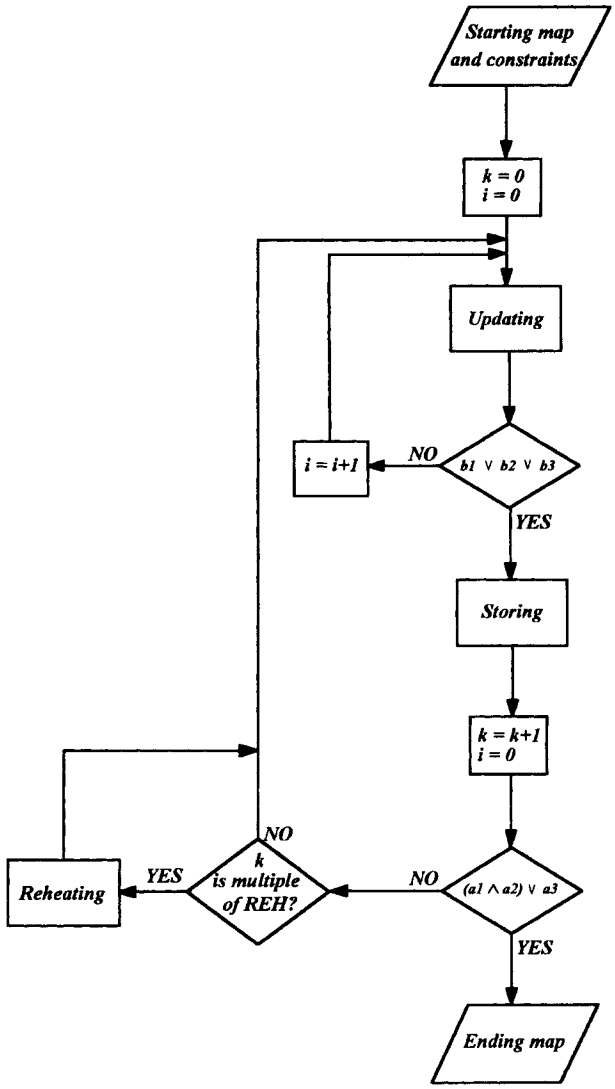

FIGURE 1. Flow chart of the PDIKF. *Updating* refers to eq. (1), *Storing* to eq. (4). *Reheating* is $C_k = C_0$ (see explanation in the text). The test conditions a1–a3 and b1–b3 are stated in Table II.

according to the following formula:

$$\mathbf{x}_{k,i+1} = \mathbf{x}_{k,0} + \mathbf{K}_{k,i}[\mathbf{z} - (h(\mathbf{x}_{k,i}) + \mathbf{H}_{k,i}(\mathbf{x}_{k,0} - \mathbf{x}_{k,i}))] \quad (1)$$

where $\mathbf{K}_{k,i}$ is a $3N \times M$ matrix, called gain matrix, defined as

$$\mathbf{K}_{k,i} = \mathbf{C}_k \mathbf{H}_{k,i}^T (\mathbf{H}_{k,i} \mathbf{C}_k \mathbf{H}_{k,i}^T + \boldsymbol{\nu})^{-1} \quad (2)$$

and $\mathbf{H}_{k,i}$ is the Jacobian $M \times 3N$ matrix of the function h defined by $\mathbf{H}_{k,i} = [\partial h(\mathbf{x})/\partial \mathbf{x}]|_{\mathbf{x}_{k,i}}$ and $h: \mathscr{R}^{3n} \rightarrow \mathscr{R}^M$ is the function that models the so-called sensor of the system. For each constraint (i.e., for each $r = 1, \ldots, M$, h_r is the rth component of the sensor function that measures the value of the geometrical property of the structure to be compared with the constraint z_r. The term h_r is a function of the state vector, and its expression depends on the type of the constraint z_r.

For distance constraints between points $\mathbf{x}_i$ and $\mathbf{x}_j$,

$$h_r(\mathbf{x}) = \|\mathbf{x}_i - \mathbf{x}_j\| \quad (3a)$$

TABLE I.
List of Symbols Used by the DIKF.

Symbol	Name	Type
N	Number of atoms	Scalar
M	Number of constraints	Scalar
$\mathbf{z} = (z_1, z_2, \ldots, z_M)$	Constraint mean vector	M-dimensional vector[a]
$\boldsymbol{\nu}$	Constraint covariance matrix	$M \times M$ matrix[a]
$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$[b]	State vector	$3N$-dimensional vector
$\mathbf{C}$	Coordinate covariance matrix	$3N \times 3N$ matrix
$\mathbf{C}_{k,i}$	$\mathbf{C}$ at the ith iteration in the kth cycle	$3N \times 3N$ matrix
h	Sensor function	Function $h: \mathbf{x} \in \mathscr{R}^{3N} \rightarrow \mathscr{R}^M$
$\mathbf{H}$	Jacobian matrix of h	$M \times 3N$ matrix
$\mathbf{H}_{k,i}$	$\mathbf{H}$ at the ith iteration in the kth cycle	$M \times 3N$ matrix
$\mathbf{K}$	Gain matrix	$3N \times M$ matrix
$\mathbf{K}_{k,i}$	$\mathbf{K}$ at the ith iteration in the kth cycle	$3N \times M$ matrix
ε	Average standard error	Scalar
μ	Maximum standard error	Scalar

[a] Constant during the computation.
[b] $\mathbf{x}_j = (x_j, y_j, z_j)$ are the coordinates of the jth atom, for $j = 1, \ldots, N$.

For bond angle constraints among points $\mathbf{x}_i$, $\mathbf{x}_j$, and $\mathbf{x}_k$ ($\mathbf{x}_j$ is the vertex),

$$h_r(\mathbf{x}) = \arccos\left((\mathbf{x}_i - \mathbf{x}_j) * (\mathbf{x}_k - \mathbf{x}_j) / \left(\|\mathbf{x}_i - \mathbf{x}_j\| \, \|\mathbf{x}_k - \mathbf{x}_j\|\right)\right) \quad (3b)$$

where $*$ is the scalar product. For dihedral angle constraints among points $\mathbf{x}_i$, $\mathbf{x}_j$, $\mathbf{x}_k$, and $\mathbf{x}_l$,

$$h_r(\mathbf{x}) = \arccos((\mathbf{T}_1 * \mathbf{T}_2)/T_1T_2) \quad \text{if} \quad (\mathbf{T}_1 \times \mathbf{T}_2) * (\mathbf{x}_i - \mathbf{x}_j) \leq 0$$

$$h_r(\mathbf{x}) = -\arccos((\mathbf{T}_1 * \mathbf{T}_2)/T_1T_2) \quad \text{if} \quad (\mathbf{T}_1 \times \mathbf{T}_2) * (\mathbf{x}_i - \mathbf{x}_j) > 0 \quad (3c)$$

TABLE II.
Ending Conditions of the PDIKF.

Conditions a	Conditions b
a1 $\varepsilon \leq \varepsilon_M$	b1 $\text{rmsd}(\mathbf{x}_{k,i+1}, \mathbf{x}_{k,i}) \leq \delta$
a2 $\mu \leq \mu_M$	b2 $\text{rmsd}(\mathbf{x}_{k,i+1}, \mathbf{x}_{k,i}) > \text{rmsd}(\mathbf{x}_{k,i}, \mathbf{x}_{k,i-1})$
a3 $k > k_M$	b3 $i > i_M$

rmsd($\mathbf{x}$, $\mathbf{y}$): Root mean square distance between the state vectors $\mathbf{x}$ and $\mathbf{y}$; ε: average standard error; μ: maximum standard error; k: cycle index; i: iteration index. ε_M, μ_M, k_M, i_M: Threshold values on the previous quantities. δ: Threshold on rmsd

where

$$\mathbf{T}_1 = (\mathbf{x}_i - \mathbf{x}_j) \times (\mathbf{x}_k - \mathbf{x}_j), \qquad T_1 = \|\mathbf{T}_1\|$$
$$\mathbf{T}_2 = (\mathbf{x}_k - \mathbf{x}_j) \times (\mathbf{x}_l - \mathbf{x}_k), \qquad T_2 = \|\mathbf{T}_2\|$$

and $\times$ is the vectorial product.

In the inner loop, cycling on the iteration index i, the starting map is iteratively updated with a simple iterated Kalman filter[9] until one of the conditions b1, b2, or b3 (see Table II) is satisfied. Condition b1 is verified when no appreciable variations of the state vector are met and, therefore, further iterations would not improve the structure. Condition b2 stops iterations in case of divergence, reducing convergence problems; it tests if the root mean square (rms) distance between two subsequently computed vectors is monotonically decreasing during the iterations, as one would expect if the map is approaching a stable region in the conformational space. Finally, b3 is a time-out condition: No more than i_M iterations are allowed. When one of these conditions is verified, one cycle of iterations has been completed; thus the covariance matrix is updated according to eq. (4) and the map is stored:

$$\begin{cases} \mathbf{x}_{k+1,0} = \mathbf{x}_{k,i} \\ \mathbf{C}_{k+1} = \mathbf{C}_k - \mathbf{K}_{k,i}\mathbf{H}_{k,i}\mathbf{C}_k \end{cases} \quad (4)$$

The exit conditions a1, a2, and a3 (see Table II) are then tested. Conditions a1 and a2 test the

fitting of the computed structure to the constraints, measuring the standard errors (average and maximum) and verifying that they are beneath preset thresholds. The average (ε) and maximum (μ) standard error (measured both in standard deviations) are calculated as

$$\varepsilon = M^{-1} \sum_{r=1}^{M} |h_r(\mathbf{x}) - z_r| \nu_{r,r}^{-1/2} \quad (5a)$$

$$\mu = \max\{|h_r(\mathbf{x}) - z_r| \nu_{r,r}^{-1/2}\}_{r=1,\ldots,M} \quad (5b)$$

where $\nu_{r,r}$ is the variance (that is, the uncertainty) of the constraint z_r, and h_r is the rth component of the sensor function.

Condition a3 is a time-out condition analogous to condition b3 (no more than k_M cycles are allowed) that prevents the program from looping when convergence is not achieved. The exit test is formulated as (a1 $\wedge$ a2) $\vee$ a3 in order to require that both average and maximum standard error be beneath the preset thresholds ε_M and μ_M. It has been verified that the trivial $\vee$ (*or*) operator between conditions a1 and a2 often led to structures satisfying only condition a2—that is, with only the maximum standard error beneath the proper threshold: $\varepsilon_M < \varepsilon \leq \mu \leq \mu_M$ [$\varepsilon \leq \mu$ is always true because of the definitions in eq. (5)]. Such structures produce unsatisfactory results because the constraints could have standard errors just under μ_M; ε could result comparable with μ, while a satisfactory structure usually has a sound value of the average standard error three times as low as the maximum one.

The reheating consists of resetting the matrix **C** to its starting value (i.e., $\mathbf{C}_k = \mathbf{C}_0$) every *REH* cycles (*REH* is a parameter definable by the user). The aim of this operation is to give the structure more freedom in moving, so if it is stranded in a region of the conformational space in which the constraints are not well satisfied, it can be displaced from there, just as if energy were given to the structure, thus raising its temperature (actually no temperature parameter is involved in the calculation).

The gain matrix $\mathbf{K}_{k,i}$ is the heart of the Kalman filter–based algorithms as nonlinear Bayesian estimators. As can be seen from its definition [eq. (2)] and from the update formula [eq. (1)], the gain matrix weighs the update of the state vector, comparing the uncertainty on the positions (matrix **C**) with the uncertainty on the experimental constraints (matrix $\boldsymbol{\nu}$). When the confidence in the constraints is lower than the confidence in the atom coordinates (i.e., the variances of the constraints in the $\boldsymbol{\nu}$ matrix are higher than the variances of the coordinates in the **C** matrix), the $(\mathbf{HCH}^T + \boldsymbol{\nu})^{-1}$ term in eq. (2) will be predominant on the first term $\mathbf{CH}^T$ and the gain matrix will have low values. Thus, uncertain constraints will be taken into little account and the updating of the state vector will be softened. In the opposite case, in which constraints are precise (low variances in $\boldsymbol{\nu}$) and few confidence is given to the coordinates (high variances in **C**), the contribution of $\boldsymbol{\nu}$ in calculating $\mathbf{K}_{k,i}$ will be negligible and the gain matrix will tend to have high values, resulting in a sensible update of the state vector.

Performance Indexes

The standard error, calculated according to eqs. (5a) and (5b), measures the fit of the computed structures to the constraints imposed, directly showing if the structure on hand belongs to the solution space. Therefore, it has been chosen as the primary indicator of the algorithm's performance.

The rms distance between the computed structure and a reference one depends on the particular choice of the latter. All structures within the conformational space which satisfy the constraints within a fixed extent are to be considered satisfactory, and some of them may have a considerable rms distance between each other. However, the rms distances between structures with low standard errors and a reference one can also be a reliable indicator of the algorithm's conformational space sampling. In fact, high rms distances computed for satisfactory structures indicate that the conformational space is well sampled by the algorithm that produced the structures, while low rms values indicate the presence of a bias that drives the algorithm in a particular region of the conformational space.

Another important issue in the case of DIKF algorithms is the reliability of the estimates provided for the atomic position variances. To this purpose, let us consider a single molecule a and a set $S^{(a)}$ of $N_{S^{(a)}}$ structures sampling its conformational space. Let $\bar{x}_i^{(a)}$ be the average of the ith atomic coordinate among all the $N_{S^{(a)}}$ structures of the molecule a—that is, the ith coordinate of the mean structure

$$\bar{x}_i^{(a)} = \frac{1}{N_{S^{(a)}}} \sum_{m=1}^{N_{S^{(a)}}} x_i^{(a,m)} \quad (6)$$

where $x_i^{(a,m)}$ is the ith atomic coordinate in the mth structure. Then the standard deviation relative to the distribution of $x_i^{(a,m)}$ is what we call the *a posteriori* standard deviation for the ith atomic coordinate in molecule a:

$$\sigma(\bar{x}_i^{(a)}) = \frac{1}{N_{S^{(a)}}} \left(\sum_{m=1}^{N_{S^{(a)}}} \left(x_i^{(a,m)} - \bar{x}_i^{(a)} \right)^2 \right)^{1/2} \quad (7a)$$

The list of $3N^{(a)}$ couples

$$\left(\bar{x}_i^{(a)}, \sigma(\bar{x}_i^{(a)}) \right)_{i=1,\ldots,3N^{(a)}},$$

where $N^{(a)}$ is the number of atoms in the amino acid a, defines what we can call the mean_map$^{(a)}$ of set $S^{(a)}$. The mean_map$^{(a)}$ can be used as an approximation of the real produced mean coordinates and variances, and its goodness will depend on how extensively the conformational space is sampled by the structures. It will be useful to our discussion to define the *a posteriori* average standard deviation for the mean_map$^{(a)}$. That is

$$\bar{\sigma}^{(a)} = \frac{1}{3N^{(a)}} \sum_{i=1}^{3N^{(a)}} \sigma(\bar{x}_i^{(a)}) \quad (7b)$$

We can summarize the estimation of the variances produced by a DIKF algorithm by two indexes, defined in analogy to the *a posteriori* parameters introduced in equations (7) for the mean_map$^{(a)}$:

- The estimated average standard deviation for the ith atomic coordinate in $N_{S^{(a)}}$ runs of a DIKF (one for each atomic coordinate $i = 1,\ldots,3N^{(a)}$), defined by

$$\bar{\sigma}_i^{(a)} = \frac{1}{N_{S^{(a)}}} \sum_{m=1}^{N_{S^{(a)}}} (\mathbf{C}^{(a,m)}(i,i))^{1/2} \quad (8a)$$

  where $\mathbf{C}^{(a,m)}$ is the covariance matrix of the mth map for the molecule a.
- The estimated average standard deviation for the map m, defined by

$$\bar{\sigma}^{(a,m)} = \frac{1}{3N^{(a)}} \sum_{i=1}^{3N^{(a)}} (\mathbf{C}^{(a,m)}(i,i))^{1/2} \quad (8b)$$

The reliability of the estimated variances can thus be assayed by comparing estimated average standard deviations and *a posteriori* ones. Since the magnitude of the estimated variances is influenced by several execution parameters,[11,12] one should not expect the *a posteriori* and estimated variances to be identical. It is important, instead, that they be well correlated and that the correlation be approximated sufficiently well by a function whose values are known at the end of a DIKF run. This issue is described in detail in the "Results" section.

The Test Set of Amino Acids

For each amino acid a, a set $S_0^{(a)}$ of $N_{S^{(a)}} = 100$ starting maps differing in the state vectors has been used. The state vectors have been generated by perturbing a correct reference† structure with a randomization protocol, ensuring that the rms distances of the perturbed structures from the reference one are uniformly distributed from 0 to 10 Å. The starting covariance matrices, equal for all starting maps of each amino acid, have been chosen to be diagonal, because in the absence of any structural information the coordinates are assumed to be independent. The diagonal elements have been set to the same value of 156.25 Å^2 that corresponds to a standard deviation of 12.5 Å, which is roughly the dimension of a residue. As distance constraints, the distances among all the atoms involved in covalent bonds or in angle bonds have been used for all the amino acids. A constraint on the dihedral angle H_α—C—C_β—N to fix chirality has been used for all the amino acids except glycine, which does not have the C_β atom. An additional dihedral angle constraint (C_α—C_{γ_1}—H_β—C_{γ_2}) has been used for isoleucine and the analogous (C_α—O_{γ_1}—H_β—C_{γ_2}) for threonine, which have an additional chiral center in their sidechain. No further dihedral angle constraints have been imposed on the sidechains; thus the conformational space compatible with the constraints included all and only the conformations compatible with the covalent geometry.

SDIKF and PDIKF have been run with the same parameters on all starting maps for each amino acid. The results have also been compared with those obtained by the optimization program Dspace 4.0 (by Hare Research, Inc.), which exploits a conjugate gradient minimization with a distance-based pseudoenergy penalty function, under the same conditions of starting structures and constraints.

As explained in the preceding paragraph, the DIKF-based algorithms allow the use of thresholds on standard errors as convergence criteria. These

† The reference structures were generated with standard covalent geometry by means of the program Insight II (rel. 2.0, by Biosym).

thresholds have been set to require that both ε and μ be lower than $2\sigma(\varepsilon_M = \mu_M = 2)$. With a typical distance constraint variance for covalent bonds of 0.01 Å^2, a threshold of 2σ means that the constraint violation should not exceed 0.2 Å. In Dspace 4.0 it is not possible to impose such a condition, because the program checks convergence only by verifying that a penalty function is zeroed and the standard errors can be evaluated only after convergence has been achieved. For this reason, the uncertainty of the constraints used in Dspace 4.0 was gradually lowered until results with average standard errors comparable with those of the DIKFs were obtained. This study resulted in using, on average, constraints with uncertainty 1000 times (1000 times in terms of variance—that is, 31.6 times in terms of standard deviations) as low for Dspace 4.0 as for the DIKFs. This means that Dspace 4.0 requires precise data to reach results comparable, in terms of average standard errors, with those obtained by the DIKFs.

Results

Table III summarizes the results obtained with the three algorithms and shows the averaged values obtained as explained in this section. Calling $\varepsilon^{(a,m)}$ and $\mu^{(a,m)}$, respectively, the average and maximum standard error for the mth map in the resulting set $S^{(a)}$, it is possible to compute their average values within each set as

$$\epsilon^{(a)} = \frac{1}{N_{S^{(a)}}} \sum_{m=1}^{N_{S^{(a)}}} \varepsilon^{(a,m)}, \qquad \mu^{(a)} = \frac{1}{N_{S^{(a)}}} \sum_{m=1}^{N_{S^{(a)}}} \mu^{(a,m)} \tag{9}$$

These quantities can be averaged among all the 20 sets $S^{(a)}$ (one for each amino acid):

$$\langle \varepsilon \rangle = \frac{1}{20} \sum_{a=1}^{20} \varepsilon^{(a)}, \qquad \langle \mu \rangle = \frac{1}{20} \sum_{a=1}^{20} \mu^{(a)} \tag{10}$$

For each algorithm, $\langle \varepsilon \rangle$ and $\langle \mu \rangle$ are shown in the first two columns of Table III. The third column of Table III shows the average values of the rms distance measured between the reference ideal structure and all the final structures, with maximum standard error below 0.50σ (we call them satisfactory structures). Within the constraint sets we used, the maximum allowed variance is 0.028 Å^2; this limit means that satisfactory structures do not violate constraints by more than 0.08 Å. As for the standard error values, the rms distances computed in each set $S^{(a)}$ have been first averaged among the satisfactory structures within each set and then averaged among the 20 sets $S^{(a)}$, giving the average value reported. The total number of satisfactory structures generated by each algorithm is reported in parentheses. The last column of Table III shows the mean number of cycles and the mean central processing unit (CPU) time spent by each algorithm to complete a run on a single structure. More detailed results are shown in Fig. 2, in which the values $\mu^{(a)}$ for the three algorithms are compared.

A first look at Table III shows that Dspace 4.0 gives the highest maximum standard errors. In fact, as can be seen from Figure 2, Dspace 4.0 often generates structures with at least one constraint not satisfied (see, for example, *arg*, *asn*, *gln*, *ile*, *lys*, and *thr* in Fig. 2, which all show $\mu^{(a)}$ over 3σ, with single maps producing $\mu^{(a,m)}$ over 14σ). The best results are offered by PDIKF, with the average $\mu^{(a)}$ (see Fig. 2) lower than σ and usually lower than the corresponding value given by the SDIKF (both the DIKFs produce maps with $\mu^{(a)}$ under 2σ, as requested by the a2 condition). These considerations lead to the conclusion that, because it is equal to the mean constraint violation among the three algorithms, PDIKF gives structures with a better stereochemistry.

As can be seen in Table III, the average rms values are similar for the three algorithms: This indicates that they span the conformational space

TABLE III.
Summarizing Averaged Results of the Tested Algorithms.

	Standard errors		rms distance of satisfactory structures	No. of cycles and CUP time per structure
	Average $\langle \varepsilon \rangle$	Maximum $\langle \mu \rangle$		
Dspace 4.0	0.127	2.042	1.20 ± 0.27 (603)	105.4 (10.5 s)
SDIKF	0.123	0.836	1.37 ± 0.25 (657)	9.5 (38 ÷ 266 s)
PDIKF	0.108	0.514	1.31 ± 0.24 (1423)	6.0 (30 ÷ 300 s)

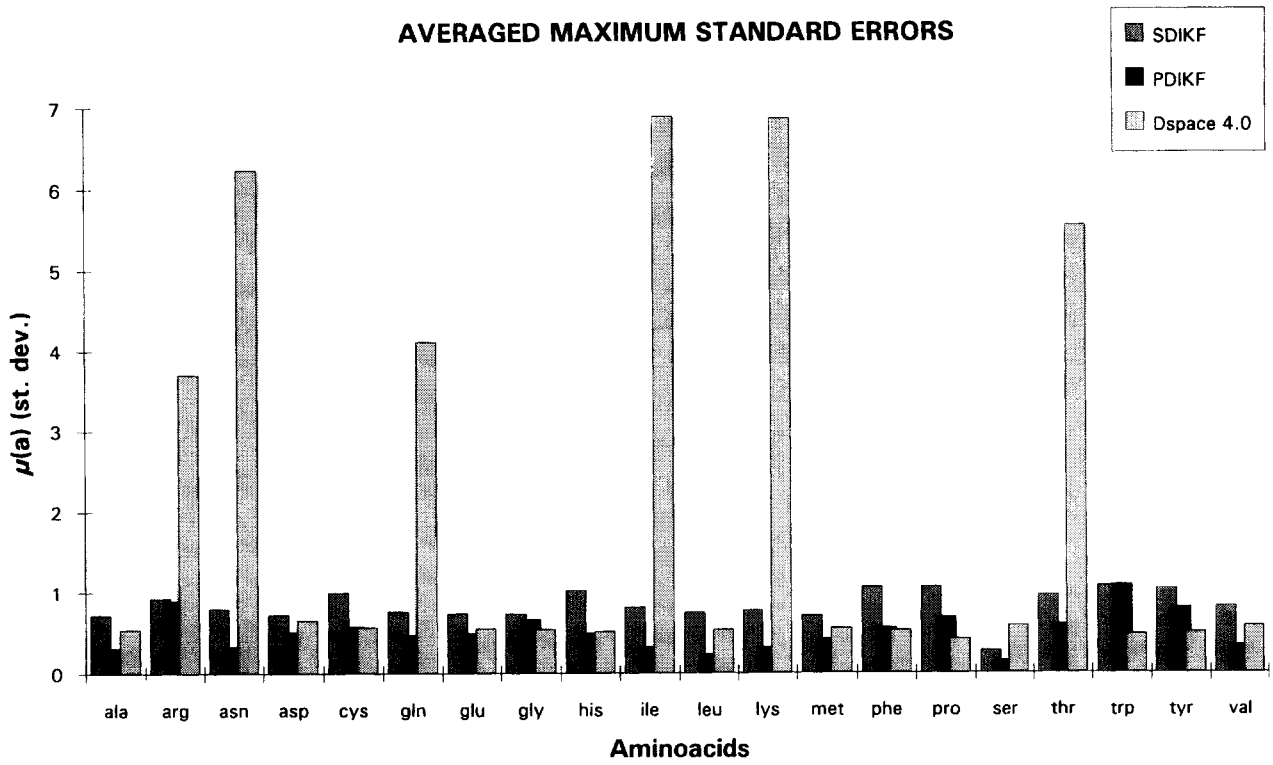

FIGURE 2. Comparison of the averaged maximum standard errors $\mu^{(a)}$ [see eq. (9)] on the final structures in each set $S^{(a)}$, obtained with the SDIKF, PDIKF, and Dspace 4.0 algorithms.

to the same extent, although PDIKF has the ability to give satisfactory results in 71% of the cases (1423 structures on a total of 2000 output structures), while Dspace 4.0 is limited to 30%.

The rms distance between the reference structure and a set of structures systematically sampling the conformational space of alanine averages to 1.09 ($\pm$0.26) Å (the conformational sampling was obtained from the reference structure, making the unconstrained dihedral angles O—C—C_α—N, C—C_α—N—$H_{N3'}$ C—C_α—C_β—$C_{\beta 3}$ vary from 0° to 340° by 20° steps). The rms distance between the reference structure and the satisfactory structures generated, respectively, by Dspace 4.0, SDIKF, and PDIKF for the same amino acid, averages to 0.96 ($\pm$0.26) Å, 1.02 ($\pm$0.32) Å, and 0.98 ($\pm$0.26) Å. Therefore, the systematically sampled set of structures gives rms values which are statistically indistinguishable from those obtained by the three algorithms, allowing us to conclude that all of them uniformly sample the solution space.

The relationships between the variances estimated by the DIKFs and the real conformational uncertainty deserve further investigation. We can make some statements. The estimated average standard deviations $\bar{\sigma}_i^{(a)}$, defined in eq. (8a), take unrealistically high values (from 2 to 20 times greater) compared to the *a posteriori* quantity $\sigma(\bar{x}_i^{(a)})$ defined in eq. (7a). However, $\bar{\sigma}_i^{(a)}$ and $\sigma(\bar{x}_i^{(a)})$ are correlated with a linear correlation coefficient ranging from 0.48 to 0.80 among all residues (see Table IV). On the basis of a single queue test between the null hypothesis of uncorrelated data and the hypothesis of positive linear correlation coefficient, all the correlation coefficients are consistent with the hypothesis of a linear correlation between estimated and *a posteriori* variances at level 0.01. This conclusion is supported by the student's t test values shown in Table IV. All these values are greater than the tabulated values of $t_{0.99}$ ($t_{0.99} \leq 2.47$, for the typical dimension of residues).[‡]

To understand this behavior better, let us recall some well-known dependencies. It has been established[11, 12] that the resulting variance estimate in-

[‡] The Student's t test values shown in Table IV have been computed as $t^{(a)} = r^{(a)}(3N^{(a)} - 2)^{1/2}(1 - r^{(a)^2})^{-1/2}$, where $r^{(a)}$ is the linear correlation coefficient and $3N^{(a)} - 2$ is the degree of freedom for the t value distribution relative to the amino acid a.

TABLE IV.
Linear Correlation Coefficients of $\bar{\sigma}_i^{(a)}$ versus $\sigma(\bar{x}_i^{(a)})$.

Amino acid	Correlation coefficient	Student's T test value
ala	0.72	6.24
arg	0.56	6.08
asn	0.70	6.79
asp	0.75	7.43
cys	0.74	6.90
gln	0.58	5.39
glu	0.79	9.32
gly	0.70	5.23
his	0.70	7.46
ile	0.79	10.25
leu	0.67	7.21
lys	0.60	6.48
met	0.56	5.18
phe	0.63	6.60
pro	0.48	3.82
ser	0.56	4.22
thr	0.78	8.38
trp	0.68	8.13
tyr	0.62	6.54
val	0.80	9.80

creases with increasing constraint and starting map variances, while it decreases with increasing number of applied constraints and with the number of cycles performed after the last reheating before the termination. Therefore, for the estimated variances to have the correct magnitude, it is not sufficient that initial variances correctly reflect the conformational uncertainty in absence of constraints and that the constraint variances be correctly estimated,[11] because the number of cycles since the last reheating also influences them. This suggests that we should investigate a correlation between a parameter that summarizes the dependencies described earlier and the ratio $\gamma^{(a,m)} = \bar{\sigma}^{(a,m)}/\bar{\sigma}^{(a)}$ between the estimated and *a posteriori* average standard deviation. The index $p^{(a,m)}$ defined next is one such parameter:

$$p^{(a,m)} = \frac{\bar{\sigma}_{\text{dist}}^{(a)}}{M_{\text{dist}}^{(a)} \cdot c^{(a,m)}} \tag{11}$$

where

$$\bar{\sigma}_{\text{dist}}^{(a)} = \frac{1}{M_{\text{dist}}^{(a)}} \sum_{r=1}^{M_{\text{dist}}^{(a)}} (V_{r,r}^{(a)})^{1/2} \tag{12}$$

is the mean standard deviation of the $M_{\text{dist}}^{(a)}$ distance constraints defined for the amino acid a, and $c^{(a,m)}$ is the number of cycles performed after the reheating before reaching the solution (for the mth map of residue a). Notably, $\bar{\sigma}_{\text{dist}}^{(a)}$, $M_{\text{dist}}^{(a)}$, and $c^{(a,m)}$ are all known at the termination of a DIKF run. In fact, $p^{(a,m)}$ is linearly correlated to $\gamma^{(a,m)}$ with the linear correlation coefficient 0.8, confirming the thesis. We can thus conclude that the final coordinate variances of the PDIKF can yield a good estimate of the real positional uncertainties if scaled by means of a coefficient obtainable by information that is available at the termination of a run.

Regarding the time performance of the algorithms, it has been found out that the SDIKF takes, on the average, from 4 to 14 cycles to reach average standard errors with a magnitude of 0.1σ, while PDIKF takes from 4 to 6 cycles. The number of cycles required by Dspace 4.0 is more than one order of magnitude bigger, from 60 to 180, depending on the number of atoms and constraints involved. Indeed, the time required to complete a cycle varies from algorithm to algorithm. On a Sun Sparc Station 1, Dspace 4.0 takes about 0.1 min of CPU per cycle, while the SDIKF takes from 2 to 28 min per cycle and the PDIKF takes from 5 to 50 min per cycle, depending on the number of atoms and constraints involved. The time required by the DIKFs to achieve convergence is one order of magnitude greater than the time required by Dspace 4.0.

These considerations hold for the present DIKF implementations—that is, without any attempt of code optimization for the heavy matricial calculations which could be carried out (e.g., exploiting sparse matrix representations). Indeed, the Jacobian matrix **H** is a sparse matrix throughout the computation: For tryptophane (27 atoms and 75 distance constraints), no less than 92.5% of its elements are zeroes, because for each constraint (i.e., for each row), no more than two atoms (i.e., six elements) are involved in the derivative of the sensor function h (the contribution of the only dihedral angle constraint is negligible; see Appendix B). For bigger compounds (such as peptides or whole proteins), this percentage is even higher and can exceed 98%. In the covariance matrix **C**, when the computation starts and after every reheating operation, only the diagonal elements are different from zero. Under the hypothesis of exploring sparse matrix representations on the matrices **H** and **C** (when possible), it has been calculated (see Appendix B) that the number of operations involved in matrix and vector calculations could be reduced up to 18% ÷ 49% of the present value, depending more on the number of cycles executed than on the number of iterations in

each cycle or on the number of atoms and constraints involved. Since it has been estimated that about 50% of the total computational time is accounted for matrix operations, it can be deduced that the code optimization with sparse matrix employment could reduce the total computational time from 60% to 75% of its present value, giving its best performance on runs with few cycles (for runs with more than seven cycles, the reduction will be greater than 71%; see Appendix B).

Another possibility of speeding up the DIKF algorithms is given by parallel computation on multiprocessor systems. We have shown that for Dspace 4.0, the complexity of the problem strongly affects the number of cycles needed to reach convergence, while for the DIKFs it more heavily affects the time required to perform a single cycle, in which the sizes of the matrices involved in calculations grow quadratically with the dimension of the problem (in terms of number of atoms and constraints). Parallel computation would speed up matrix calculation within the single cycle, and it is thus predictable that the implementation on a parallel computer of the PDIKF, which effectively exploits the parallel application of the constraints and which gives better results than SDIKF, would improve its speed performance.

Conclusions

The performance analysis of the reconstruction of amino acidic structures carried out in this work has shown that DIKF-based algorithms (PDIKF in particular) are well suited for determination of the three-dimensional structures of small molecules based on knowledge of their interatomic distances. All the amino acidic structures have been reconstructed within the preset thresholds on standard errors and present, on average, a better stereochemistry if compared to the structures obtained with the optimization program used for the comparison.

In our evaluation, the possibility of defining termination conditions based directly on indicators of constraint satisfaction (average and maximum standard error) has proven useful for obtaining high-quality structures and for verifying the best quality of structures obtainable with a given algorithm. Many commercially available and academic programs do not allow termination conditions based on the degree of constraint satisfaction, and this lack sets artificial limits on the quality of the structures obtainable from them. The limits are artificial in the sense that they are not intrinsic to the optimization algorithm itself but depend on the code encapsulating it. In particular, Dspace 4.0 presents this shortcoming, and this plays a role in the performance difference with the DIKFs. We had to overcome this difficulty by artificially lowering constraint uncertainties. In a number of cases, Dspace 4.0 did not achieve the same degree of satisfaction obtained by the DIKFs (in particular, by PDIKF; see Fig. 2). Thus, the better average quality of structures generated with the PDIKF is due to a superiority of the PDIKF algorithm implemented here over the conjugate gradient algorithm with harmonic penalty function, rather than to limitations imposed by the code encapsulating the minimization algorithm in Dspace 4.0.

Furthermore, the DIKF-based algorithms guarantee more complete data processing with a full handling of the uncertainty of the experimental data (the constraints) and of the structure geometry (the coordinates). The algorithm is able to provide an estimate of the uncertainty on atomic positions at each step of the computation, while other methods would need a large number of trials to build statistics on the atomic coordinates.

The limitation of the algorithm lies essentially in its heavy matrix calculations and its demand of computational resources. Sparse matrix algorithms or parallel computation on multiprocessor systems are likely to reduce this problem, because they would speed up the matrix calculation within the single cycle. In fact, we have shown that an increase in the complexity of the problem at hand would not increase significantly the number of cycles needed to reach convergence; rather, it heavily affects the computing time required by each cycle. Tests performed with sparse matrix algorithms and on parallel machines may result to determine the real applicability of the Kalman filter on more complex structures, such as proteins.

Acknowledgments

Part of this work was supported by a research contract from TECHNOBIOCHIP (Marciana, Italy) and FARMITALIA CARLO ERBA (Milano, Italy) within the framework of the National Program Technology for Bioelectronics, sponsored by the Italian Ministry of Universities, Scientific and Technological Research.

Appendix A

PARALLEL VERSUS SEQUENTIAL DIKF

The sequential DIKF algorithm (SDIKF) differs from the parallel one (PDIKF), described in the text, in the following: The constraints are not all applied simultaneously, but one at a time; the reheating is not parameterized (it is performed at every cycle), and condition b2 is not applied. In this appendix we show that the parallel computation is not equivalent to the sequential one, regardless of the reheating parameterization and condition b2. For this purpose, it is sufficient to show that there exists at least a single case in which the two algorithms give different results after a single iteration ($k_M = 1, i_M = 1$).

The simplest nontrivial case consists of $N = 3$ points and $M = 2$ distance constraints. With only one distance constraint, the gain matrix and the Jacobian matrix reduce to vectors; in this case the two algorithms become equivalent because the matricial calculations reduce to a single vector operation, precisely the same as that exploited by the SDIKF.

The starting map we chose is an isosceles triangle T with the basis AB (2 units long) along the x-axis, 1 unit height, and the vector C on the y-axis; explicitly:

$$T = \{A(0,1,0); B(-1,0,0); C(1,0,0)\} \quad \text{(A.1)}$$

As constraints, we imposed the length of the two equal sides of the triangle ($\sqrt{2}$ units long) to be precisely (i.e., with zero variance) 1 unit long. The constraints are stored in a list from which the $\mathbf{z}$ vector takes its values:

$$\mathbf{z} = \begin{pmatrix} d(A,C) \\ d(B,C) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \text{(A.2)}$$

The starting covariance matrix has been chosen to be diagonal—that is $\mathbf{C} = \alpha\mathbf{I}$, where $\mathbf{I}$ is the identity matrix and α is an arbitrary value (we set it to 156.25, the same value used in the tests described in the text).

As can be verified by applying the algorithms by hand, SDIKF and PDIKF give the following different results (rounded to the third decimal digit):

$$\text{SDIKF: } T_{\text{Seq}} = \{A(0.016, 0.714, 0); B(-0.842, 0.158, 0); C(0.827, 0.129, 0)\} \quad \text{(A.3)}$$

$$\text{PDIKF: } T_{\text{Par}} = \{A(0, 0.707, 0); B(-0.854, 0.146, 0); C(0.854, 0.146, 0)\} \quad \text{(A.4)}$$

Reversing the order of the constraints in the constraints list, the results are as follows:

$$\text{SDIKF: } T'_{\text{Seq}} = \{A(-0.016, 0.714, 0); B(-0.827, 0.129, 0); C(0.842, 0.158, 0)\} \quad \text{(A.5)}$$

$$\text{PDIKF: } T'_{\text{Par}} \equiv T_{\text{Par}} \quad \text{(A.6)}$$

The differences between the coordinate values of T_{Seq} and T_{Par} (up to 0.027 units—that is, around 3% of the coordinate value) are due to the different mode of calculation in the two algorithms, as we explain later.

The PDIKF, for each iteration, exactly performs what is stated in the updating formula of eq. (1) (see text)—that is, the constraint vector $\mathbf{z}$ is compared with the correspondent measures taken on the structure (sensor function h), shifted by the Jacobian matrix contribution. The difference between these quantities is then weighted by the gain matrix $\mathbf{K}$, and finally the coordinate vector is updated.

SDIKF, instead, starts considering the first constraint in the vector $\mathbf{z}$; it iteratively applies the updating formula of eq. (1) as if that constraint were the only one. This causes $\mathbf{z}$ to be now a scalar entity, and h becomes a scalar too, while $\mathbf{H}$ contains the first row of the actual $\mathbf{H}$ matrix; the gain matrix $\mathbf{K}$ is also reduced to its first column. After the single requested iteration has been competed, the covariance matrix $\mathbf{C}$, the sensor function h, its Jacobian $\mathbf{H}$, and the gain matrix $\mathbf{K}$ are recomputed on the just obtained intermediate coordinates; the second constraint is picked up, and a new series of iterations (only one in our example) starts on the second constraint only.

We can thus conclude that the parallel DIKF is not equivalent to the sequential one.

Appendix B

ESTIMATION OF MATRIX OPERATION REDUCTION EXPLOITING SPARSE MATRIX REPRESENTATION IN THE PDIKF ALGORITHM

Let N be the number of atoms (for a total of $3N$ coordinates) and M the number of constraints. The number of multiplications (P) and sums (S) between floating point numbers in each PDIKF

cycle may be written as

$$\begin{cases} P = P_{\text{cycle}} + rP_{\text{iter}} \\ S = S_{\text{cycle}} + rS_{\text{iter}} \end{cases} \tag{B.1}$$

where $P_{\text{cycle}}(S_{\text{cycle}})$ are the number of multiplications (sums) carried out for each cycle (independent of the number of iterations involved) and $P_{\text{iter}}(S_{\text{iter}})$ is the number of multiplications (sums) executed in each iteration; r is the number of iterations within the cycle, whose value (ranging between 1 and 10) depends on conditions b1 and b2 (see Table II).

$P_{\text{cycle}}(S_{\text{cycle}})$ effectively represents the number of multiplications (sums) required to compute the covariance matrix **C** at the end of every cycle in eq. (4): We then call it $P_C(S_C)$. $P_{\text{iter}}(S_{\text{iter}})$ is built up by the contributions P_x and P_K (S_x and S_K) of multiplications (sums) necessary to compute the state vector **x**, in eq. (1), and the reduction matrix **K**, in eq. (2). That is,

$$\begin{cases} P_{\text{cycle}} \equiv P_C \\ S_{\text{cycle}} \equiv S_C \end{cases} \qquad \begin{cases} P_{\text{iter}} = P_x + P_K \\ S_{\text{iter}} = S_x + S_K \end{cases} \tag{B.2}$$

In the present implementation of the PDIKF algorithm, the operations on matrices are carried out element by element, with CPU time waste if the elements involved are zeroes. Indeed, the matrices **H** and **C** may be considered sparse matrices under the hypothesis shown later.

The matrix **H** (see Table I) consists of M rows and $3N$ columns. For each row (i.e., for each constraint), each element corresponding to a coordinate not involved in the derivative of the sensor function is set to zero. That is, in the rows corresponding to distance constraints, only six nonzero elements can be present, because only two atoms are involved in the constraint; similarly, only 9 and 12 nonzero elements are allowed, respectively, for bond angle constraints (three atoms involved) and dihedral angle constraints (four atoms involved). Splitting M into its different constraint type components—that is, M_d distance constraints, M_{bond} bond angle constraints, and M_{dih} dihedral angle constraints ($M = M_d + M_{\text{bond}} + M_{\text{dih}}$)—the minimum number of nonzero elements in the matrix **H** can be computed as $6M_d + 9M_{\text{bond}} + 12M_{\text{dih}}$, on a total of $3NM$ elements. It follows that, by using a suitable sparse matrix codification, the operations on the matrix **H** can be reduced at least by a factor $\alpha = (2M_d + 3M_{\text{bond}} + 4M_{\text{dih}})/NM$.

The matrix **C** consists of $3N$ rows and $3N$ columns. When the computation starts, and after every reheating operation (we call these cases special cycles), only the $3N$ diagonal elements are nonzero. Thus, in the special cycles, the operations on the matrix **C** can be reduced by a factor $\beta = 1/3N$. It follows that when the matrices **H** (or $\mathbf{H}^T$) and **C** are multiplied between them, the operations can be reduced by a factor $\alpha\beta$ in the special cycles and by a factor α in the other cases. When the matrix $\mathbf{CH}^T$ is involved in multiplications, it contributes to the reduction by a factor α in the special cycles (because if **C** is diagonal, $\mathbf{CH}^T$ conserves the degree of sparseness of $\mathbf{H}^T$—that is, the number of nonzero elements does not change), and no reduction can be predicted in the other cases (because the product of a matrix with a sparse matrix is not necessarily a sparse matrix).

Introducing the coefficients

$$\delta_\alpha = \begin{cases} \alpha \text{ (sp.cyc.)} \\ 1 \text{ (others)} \end{cases} \qquad \delta_\beta = \begin{cases} \beta \text{ (sp.cyc.)} \\ 1 \text{ (others)} \end{cases} \tag{B.3}$$

we can then write the contributions in eq. (B.2) as

$$\begin{cases} P_C = \underbrace{\alpha\delta_\beta 9N^2M}_{HC} + \underbrace{\delta_\alpha 9N^2M}_{K(HC)} \\ P_x = \underbrace{\alpha 3NM}_{H\Delta x} + \underbrace{3NM}_{K[z-(h+H\Delta x)]} \\ P_k = \underbrace{\alpha\delta_\beta 9N^2M}_{CH^T} + \underbrace{\alpha\delta_\alpha 3NM^2}_{HCH^T} + \underbrace{M^3}_{(HCH^T+\nu)^{-1}} \\ \qquad + \underbrace{\delta_\alpha 3NM^2}_{CH^T(HCH^T+\nu)^{-1}} \end{cases} \tag{B.4}$$

$$\begin{cases} S_C = P_C + \underbrace{\delta_\beta 3N}_{C-KHC} \\ S_x = P_x + \underbrace{3N}_{\Delta x} + \underbrace{M}_{h-H\Delta x} + \underbrace{M}_{z-(h-H\Delta x)} \\ \qquad + \underbrace{3N}_{x+K(z-(h-H\Delta x))} \\ S_K = P_K + \underbrace{M}_{HCH^T+\nu} \end{cases} \tag{B.5}$$

where each single contribution is subscribed by the equation to which it refers (for simplicity, the cycle and iteration indexes k and i have been omitted, and the difference $\mathbf{x}_{k,0} - \mathbf{x}_{k,i}$ has been

TABLE B.I.
The Time Reduction Factor *R* (in %) for Glycine and Tryptophane in a PDIKF Cycle with *r* Iterations.

	Special cycles			Other cycles		
	$r = 1$	$r = 5$	$r = 10$	$r = 1$	$r = 5$	$r = 10$
Gly	19.81	24.03	24.84	56.85	55.44	55.18
Trp	18.10	23.57	24.57	52.80	52.43	52.37

replaced by Δx).[§] Grouping these contributes, and labeling P and S with the values for α and β for which they have been computed, we can finally rewrite the eq. (B.1) as

$$\begin{cases} P_{\alpha,\beta} = (\alpha\delta_\beta + \delta_\alpha)9N^2M + rM\left[\alpha\delta_\beta 9N^2 \right. \\ \qquad \left. +(\alpha + 1)(\delta_\alpha M + 1)3N + M^2\right] \\ S_{\alpha,\beta} = P_{\alpha,\beta} + 3\left[\delta_\beta N + r(2N + M)\right] \end{cases} \tag{B.6}$$

If no code optimization for sparse matrices is used, the coefficients α and β (and then δ_α and δ_β, no matter if a normal or special cycle is concerned) are equal to 1, and then eq. (B.6) becomes

$$\begin{cases} P_{1,1} = 18N^2M + rM[(3N + M)^2 + 6N] \\ S_{1,1} = P_{1,1} + 3[N + r(2N + M)] \end{cases} \tag{B.7}$$

Given that on the computer on which the tests have been performed (a Sun Sparc Station 1) there is no appreciable difference (less than 4%) in the time required to carry out a floating point sum or multiplication, the ratio

$$R = \frac{P_{\alpha,\beta} + S_{\alpha,\beta}}{P_{1,1} + S_{1,1}} \tag{B.8}$$

will give an estimation of the computational time reduction per cycle on matrix operations, obtainable by the employment of sparse matrix representation.

To get an idea of the order of magnitude of such a reduction, we can compute R for the simplest and the most complex amino acid—that is, glycine ($N = 10$, $M = 23$,$\alpha = 0.200$, $\beta = 0.033$) and tryptophane ($N = 27$, $M = 76$, $\alpha = 0.075$, $\beta = 0.012$), for $r = 1, 5, 10$. The results are summarized in Table B.1. Considering all the hypothetical runs of length ranging from 1 to 100 cycles, each cycle being built up by a fixed number r of iterations ($r = 1, 5, 10$), and supposing a reheating operation every five cycles, it can be computed that the average reduction in each run would range from 20 to 49% for Gly and from 18% to 47% for Trp; in both cases, the average reduction is over 44% for runs longer than seven cycles.

[§] The M^3 term in the P_K expression of eq. (B.4) has no reduction factor because no sparse matrix representation has been taken into account for the inversion of matrices; only M sums are considered in the S_K expression of eq. (B.5) because, as far as the tests in this work are concerned, a diagonal constant covariance matrix ν has been assumed.

References

1. M. Nigels, G. M. Clore, and A. M. Gronenborg, *FEBS Lett.*, **229**, 317 (1988).
2. G. M. Clore, A. T. Brünger, M. Karplus, and A. M. Gronenborg, *J. Mol. Biol.*, **191**, 523 (1986).
3. W. Braun, C. Bösch, L. R. Brown, N. Gô, and K. Wüthrich, *Biochimica et Biophysica Acta*, **667**, 377 (1981).
4. W. Braun and N. Gô, *J. Mol. Biol.*, **186**, 611 (1985).
5. J. F. Brinkley, R. B. Altman, B. S. Duncan, B. Buchanan, and O. Jardetzky, *J. Chem. Inf. Comp. Sci.*, **28**, 194 (1988).
6. D. A. Bassalino, F. Hirata, D. B. Kitchen, D. Kominos, A. Pardi, and R. M. Levy, *Int. J. Supercomput. Appl.*, **2**, 41 (1988).
7. W. J. Metzler, D. R. Hare, and A. Pardi, *Biochemistry*, **28**, 7045 (1989).
8. A. Gelb, *Applied Optimal Extimation*, MIT Press, Cambridge, MA, 1984.
9. R. B. Altman and O. Jardetzky, *Meth. Enzymol.*, **177**, 218 (1989).
10. R. Pachter, R. B. Altman, and O. Jardetzky, *J. Magn. Reson.*, **89**, 578 (1990).
11. R. Pachter, R. B. Altman, J. Czaplicki, and O. Jardetzky, *J. Magnet. Reson.*, **92**, 468 (1991).
12. E. A. Carrara, In *SIF: Conference Proceedings*, Vol. 31, *Cybernetics and Biophysics Italian Conference*, C. Frediani, Ed., Bologna, Italy, 1991, p. 15.
13. P. Koeh, J. F. Lefèvre, and O. Jardetzky, *J. Mol. Biol.*, **223**, 299 (1992).
14. C. C. Chen, J. P. Singh, W. B. Poland, and R. B. Altman, *Parallel Protein Structure Determination from Uncertain Data, Proceedings of Supercomputing '94*, Washington, DC, November 14–18, 1994.